

7.1查找的基本概念

基本概念

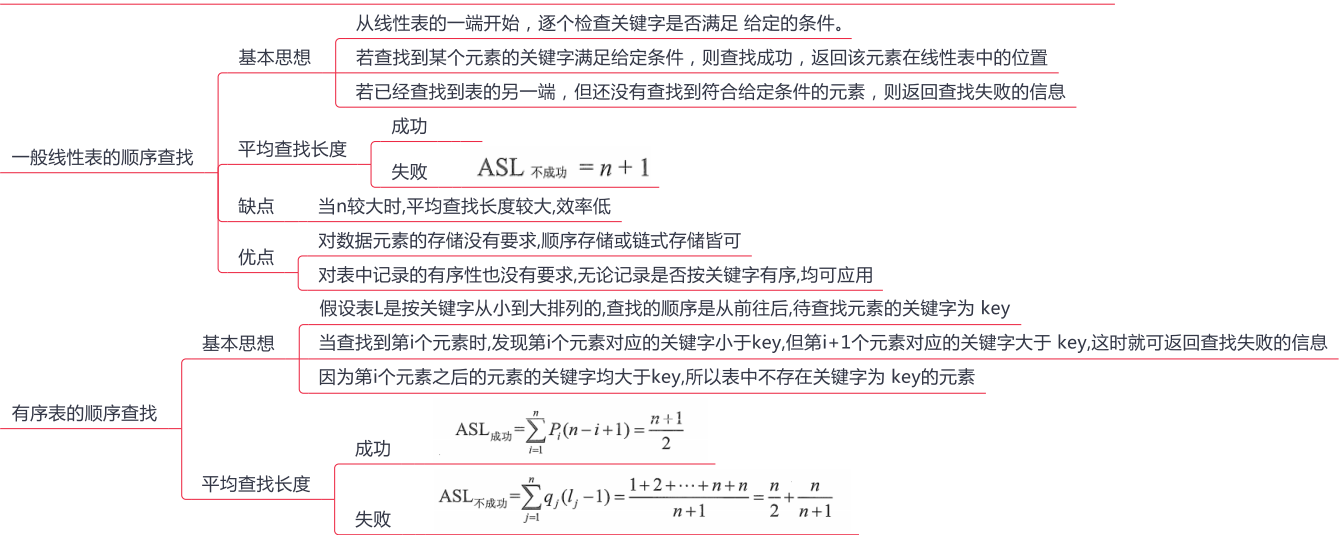
	查找：在数据集中寻找满足某种条件的数据元素的过程称为查找
	查找表（查找结构）：用于查找的数据集合称为查找表，它由同一类型的数据元素（或记录）组成，可以是一个数组或链表等数据类型
查找表操作	查询某个特定的数据元素是否在查找表中
	检索满足条件的某个特定的数据元素的各种属性
	在查找表中插入一个数据元素
	从查找表中删除某个数据元素
静态查找表	查找后不会对表进行任何修改
动态查找表	查找后对表进行修改
	关键字：数据元素中唯一标识该元素的某个数据项的值，使用基于关键字的查找，查找结果应该是唯一的
平均查找长度	在查找过程中，一次查找的长度是指需要比较的关键字次数，而平均查找长度则是所有查找过程中进行关键字的比较次数的平均值
	数学表达式 $ASL = \sum_{i=1}^n P_i C_i$
	参数含义 <div>n是查找表的长度</div>
	<div>P是查找第i个数据元素的概率,一般认为每个数据元素的查找概率相等,即P=1/n</div> <div>Ci是找到第i个数据元素所需进行的比较次数</div>

平均查找长度是衡量查找算法效率的最主要的指标

7.2顺序查找和折半查找

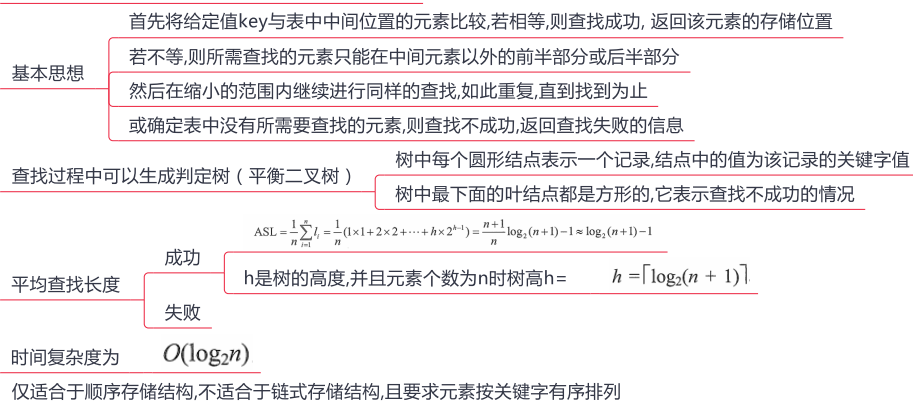
顺序查找

又称线性查找，主要用于在线性表中进行查找。顺序查找通常分为对一般的无序线性表的顺序查找和对按关键字有序的顺序表的顺序查找



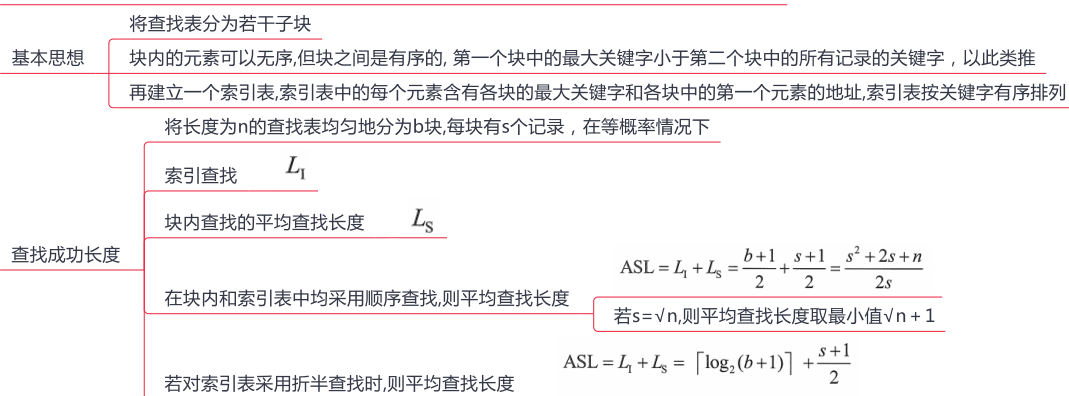
折半查找

折半查找又称二分查找，它仅适用于有序的顺序表

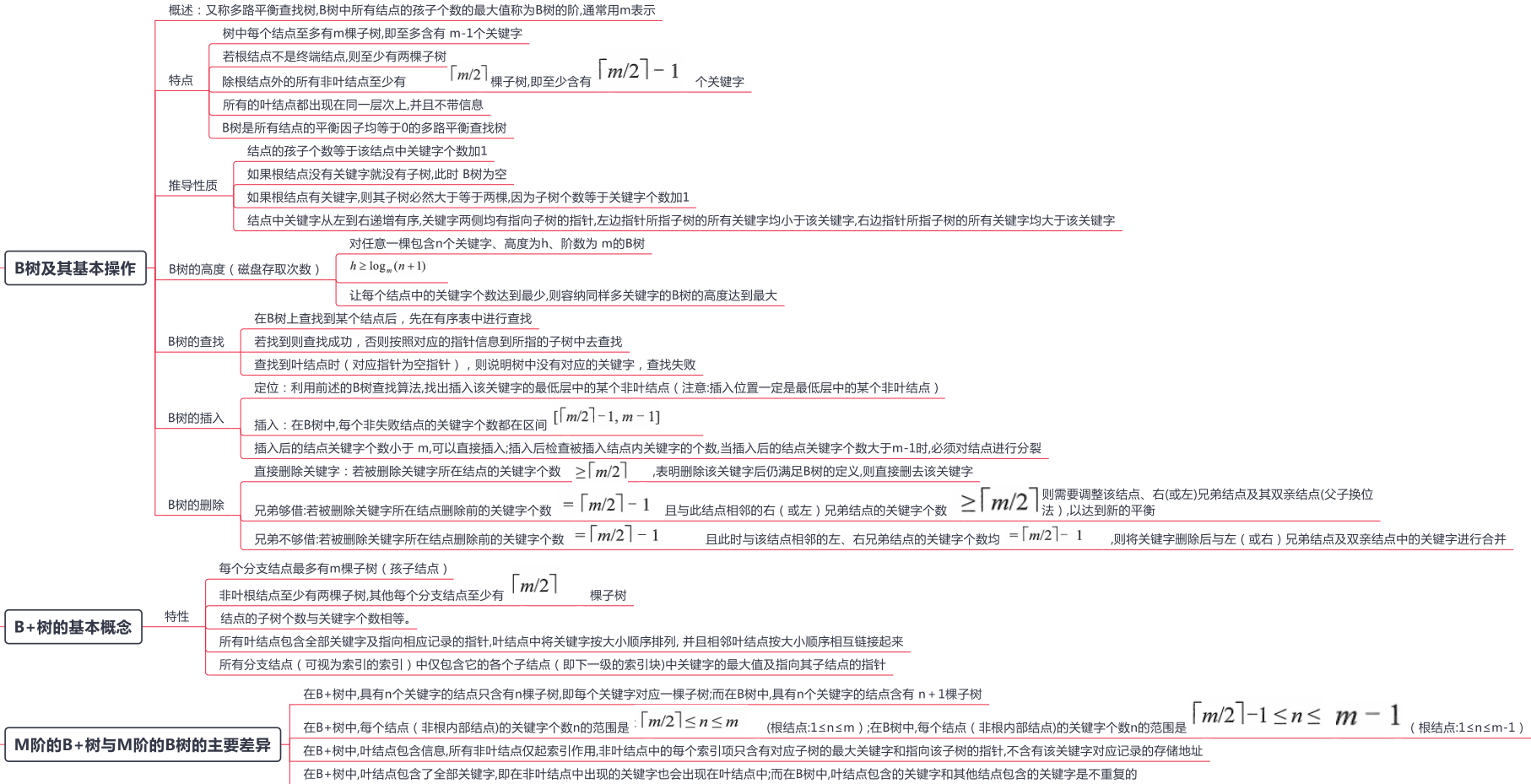


分块查找

又称索引顺序查找，它吸取了顺序查找和折半查找各自的优点，既有动态结构，又适于快速查找



7.3B树和B+树



M阶的B+树与M阶的B树的主要差异

在B+树中,具有n个关键字的结点只含有n棵子树,即每个关键字对应一棵子树;而在B树中,具有n个关键字的结点含有 n+1棵子树

在B+树中,每个结点（非根内部结点）的关键字个数n的范围是： $\lceil m/2 \rceil \leq n \leq m$ （根结点:1≤n≤m）;在B树中,每个结点（非根内部结点）的关键字个数n的范围是 $\lceil m/2 \rceil - 1 \leq n \leq m - 1$ （根结点:1≤n≤m-1）

在B+树中,叶结点包含信息,所有非叶结点仅起索引作用,非叶结点中的每个索引项只含有对应子树的最大关键字和指向该子树的指针,不含有该关键字对应记录的存储地址

在B+树中,叶结点包含了全部关键字,即在非叶结点中出现的关键字也会出现在叶结点中;而在B树中,叶结点包含的关键字和其他结点包含的关键字是不重复的

7.4散列表（上）

散列表的基本概念

- 散列函数:一个把查找表中的关键字映射成该关键字对应的地址的函数,记为 $\text{Hash}(\text{key}) = \text{Addr}$
- 冲突：散列函数可能会把两个或两个以上的不同关键字映射到同一地址
- 散列表:根据关键字而直接进行访问的数据结构
- 对散列表进行查找的时间复杂度为 $O(1)$

散列函数的构造方法

注意

- 散列函数的定义域必须包含全部需要存储的关键字，而值域的范围则依赖于散列表的大小或地址范围
- 散列函数计算出来的地址应该能等概率、均匀地分布在整个地址空间中，从而减少冲突的发生
- 散列函数应尽量简单，能够在较短的时间内计算出任一关键字对应的散列地址

散列函数

直接定址法

- 直接取关键字的某个线性函数值为散列地址
- 散列函数为 $H(\text{key}) = \text{key}$ 或 $H(\text{key}) = a * \text{key} + b$ a 和 b 是常数
- 特点
 - 计算最简单,且不会产生冲突
 - 适合关键字的分布基本连续的情况
 - 若关键字分布不连续,空位较多,则会造成存储空间的浪费

除留余数法

- 假定散列表表长为 m ,取一个不大于 m 但最接近或等于 m 的质数 p ,利用以下公式把关键字转换成散列地址
- 散列函数为 $H(\text{key}) = \text{key} \% p$
- 特点
 - 最简单、最常用的方法
 - 关键是选好 p ,使得每个关键字通过该函数转换后等概率地映射到散列空间上的任一地址,从而尽可能减少冲突的可能性

数字分析法

- 设关键字是 r 进制数,而 r 个数码在各位上出现的频率不一定相同,可能在某些位上分布均匀一些,每种数码出现的机会均等
- 而在某些位上分布不均匀,只有某几种数码经常出现,此时应选取数码分布较为均匀的若干位作为散列地址
- 特点
 - 适合于已知的关键字集合,若更换了关键字
 - 则需要重新构造新的散列函数

平方取中法

- 取关键字的平方值的中间几位作为散列地址
- 特点
 - 散列地址分布比较均匀
 - 适用于关键字的每位取值都不够均匀或均小于散列地址所需的位数

7.4散列表（下）

处理冲突的方法

开放定址法

数学递推公式

$$H_i = (H(\text{key}) + d_i) \% m$$

H (key) 为散列函数

m表示散列表表长;d_i为增量序列

增量的取值方法

线性探测法

$$d_i = 0, 1, 2, \dots, m - 1$$

当出现冲突时, 就会顺序的向下一个单元探测, 直到单元没有发生冲突

优点: 简单 容易实现

缺点: 会出现聚集现象, 降低查找效率

平方探测法

$$d_i = 0^2, 1^2, -1^2, 2^2, -2^2, \dots, k^2, -k^2$$

优点: 可以避免出现"堆积"问题,

缺点: 不能探测到散列表上的所有单元,但至少能探测到一半单元

再散列法

$$d_i = \text{Hash}_2(\text{key})$$

当通过第一个散列函数H(key)得到的地址发生冲突时,则利用第二个散列函数Hash(key)计算该关键字的地址增量

$$\text{计算公式公式 } H_i = (H(\text{key}) + i \times \text{Hash}_2(\text{key})) \% m$$

伪随机序列法

$d_i =$ 伪随机数序列时,称为伪随机序列法

拉链法

把所有的同义词存储在一个线性链表中,这个线性链表由其散列地址唯一标识

所有同义词就像被拉链串在一起一样

散列查找及性能分析

实现过程

①检测查找表中地址为Addr的位置上是否有记录,若无记录, 返回查找失败; 若有记录, 比较它与key的值, 若相等, 则返回查找成功标志, 否则执行步骤②

②用给定的处理冲突方法计算"下一个散列地址", 并把Addr置为此地址, 转入步骤①

特点

平均查找长度作为衡量散列表的查找效率的度量

散列表的查找效率取决于三个因素:散列函数、处理冲突的方法和装填因子

$$\alpha = \frac{\text{表中记录数 } n}{\text{散列表长度 } m}$$

装填因子

装填因子越大越容易发生冲突